Docket No.: POU920030116US1

Inventor: Browne et al.

Title: METHOD AND SYSTEM FOR MANAGING DATA ACCESS REQUESTS UTILIZING STORAGE META DATA PROCESSING

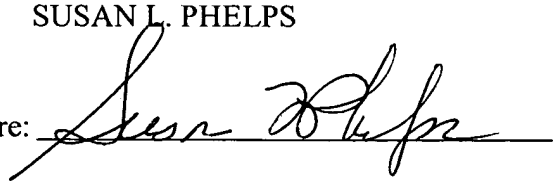
APPLICATION FOR UNITED STATES

LETTERS PATENT

"Express Mail" Mailing Label No.: ER 046554995 US
Date of Deposit: December 31, 2003

I hereby certify that this paper is being deposited with the United States Postal Service as "Express Mail Post Office to Addressee" service under 37 CFR 1.10 on the date indicated above and is addressed to: Mail Stop PATENT APPLICATION, Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.

Name: SUSAN L. PHELPS

Signature: 

INTERNATIONAL BUSINESS MACHINES CORPORATION

METHOD AND SYSTEM FOR MANAGING DATA ACCESS REQUESTS UTILIZING STORAGE META DATA PROCESSING

Technical Field

[0001] This invention relates, in general, to the management of requests to access data in communications environments, and more particularly, to informing a data object manager of an anticipated request to access the data from storage media based on a received request associated with meta data which corresponds to the data.

Background of the Invention

[0002] Storage subsystems generally consist of a number of disk drives that can be aggregated and made to appear as virtual disk drives to one or more client computers. To improve performance, storage subsystems usually deploy a cache which is used to hold frequently accessed disk blocks. The choice of which disk blocks to cache can have a significant impact on overall system performance. Some storage subsystems attempt to anticipate which disk blocks may be required by client computers by examining historical patterns of access to disk blocks. The nature of such cache management algorithms is predictive.

[0003] Although there are techniques today for the management of requests to access data in communications environments, these techniques can cause a storage subsystem to load data into its cache that is not accessed within the expected time because of their predictive nature. Thus, there is still a need for further techniques to facilitate the management of requests to access data in computer environments.

Summary of the Invention

[0004] The shortcomings of the prior art are overcome and additional advantages are provided through the provision of a method of managing requests. In one aspect, a manager receives a request associated with meta data corresponding to data maintained

separately from the meta data. In another aspect of the present invention, the manager informs another manager of an anticipated request that will be received by the another manager to enable it to prepare for the anticipated request.

[0005] Systems and computer program products corresponding to the above-summarized methods are also described and claimed herein.

[0006] Additional features and advantages are realized through the techniques of the present invention. Other embodiments and aspects of the invention are described in detail herein and are considered a part of the claimed invention.

Brief Description of the Drawings

[0007] The subject matter which is regarded as the invention is particularly pointed out and distinctly claimed in the claims at the conclusion of the specification. The foregoing and other objects, features, and advantages of the invention are apparent from the following detailed description taken in conjunction with the accompanying drawings in which:

[0008] FIG. 1 illustrates a flowchart of one embodiment of a technique for managing requests associated with data in a computer environment, in accordance with an aspect of the present invention;

[0009] FIG. 2 illustrates a flowchart of another embodiment of a technique for managing requests associated with data in a computer environment, in accordance with an aspect of the present invention;

[0010] FIG. 3 illustrates one embodiment of a technique for managing requests for access to data in an environment in which data and meta data associated with the data are stored separately, in accordance with an aspect of the present invention;

[0011] FIG. 4 illustrates an example of an environment in which a technique for managing requests for access to data is utilized, in accordance with an aspect of the present invention;

[0012] FIG. 5 illustrates another example of an environment in which a technique for managing requests for access to data is utilized, in accordance with an aspect of the present invention; and

[0013] FIG. 6 illustrates a third example of an environment in which a technique for managing requests for access to data is utilized, in accordance with an aspect of the present invention.

Best Mode for Carrying Out the Invention

[0014] In one aspect of the present invention, a manager receives a request associated with meta data. The manager informs another manager of an anticipated request to be received the another manager to enable the another manager to prepare for the anticipated request.

[0015] A technique for managing requests associated with data in a computer environment in accordance with an aspect of the present invention is described below with reference to request management flowchart 60 illustrated in FIG. 1. First, step 61 comprises a request manager receiving a request associated with meta data. Then, in step 62, the request manager informs a data object manager of a change in a data object's meta data. The data object manager makes a data object management decision in step 63 and, if necessary, acts to implement the data object management decision in step 64.

[0016] Further aspects of a technique for managing requests associated with data objects in a computer environment in accordance with the present invention are described below with reference to flowchart 50 illustrated in FIG. 2. First, step 51 comprises a request manager receiving a usage request. Then, if the communications unit is granted permission to use a data object as determined in step 52, a request manager sends a

request-management message to a data object manager and responds, substantially simultaneously, to a communications unit with a usage-request response in steps 53 and 55, respectively. Subsequent to steps 53 and 55, respectively, the data object manager prepares for an anticipated request in step 54, and data-usage communications are transmitted between the communications unit and data object manager in step 56. In one example, the step of transmitting data-usage communications includes transmitting by a communications unit requests for data blocks and transmitting by a data object manager data comprising the requested data blocks of the data object. Alternatively, if the communications unit is not granted permission to use a data object, step 57 comprises the request manager responding to a communications unit with a usage-request response.

[0017] One example of a communications environment in which a technique for managing requests associated with data objects is utilized in accordance with an aspect of the present invention is described below with reference to FIG. 3. In an environment in which the meta data associated with data objects may be stored separately from the data objects, a request manager 10 receives usage requests from a communications unit 20 via request management network 12. Request manager 10 transmits a request-management message to a data object manager 30 via a private network 14 and responds to communications unit 20 with a usage-request response via request management network 12. If communications unit 20 is granted permission to use a data object 40, the communications to support use of data object 40 are transmitted via data network 16 between communications unit 20 and data object manager 30.

[0018] Generally, both meta data and user data are associated with data objects in a computer communications environment. User data is the information that has meaning to a user or to a program that may process that data. Examples of user data are the contents of a Freelance Graphics® presentation, or employee information stored within a relational database. Meta data is information about user data. Examples of meta data associated with data objects include the identity of client computers that have access permission, data object type, names of files associated with a set of disk blocks, the

length of a file, the list of blocks that constitute a file, information about user access permissions, and the date and time a file has been created or updated. Data objects comprise data. Data object types include data files, checkpoint files, file systems, logical volumes, and journaled file system (JFS) logical volume logs.

[0019] Features facilitated by use of the technique in the computer environment illustrated in FIG. 3 include: improved security with respect to access to data objects, regulation of the speed of access to data objects, arbitration of access priorities to data objects, and increased speed of access to data objects.

[0020] An emerging class of storage environments separates the storage of user data and meta data and provides separate networks over which the user data and meta data traverse. An example of such a storage environment is IBM's Storage Tank™ file system wherein a Storage Tank™ client (a computer) accesses user data from a storage subsystem (over a storage area network (SAN) using block transfer protocols) and accesses meta data from a centralized Storage Tank™ meta data controller (over Ethernet using TCP/IP protocols). The separation of user data and meta data can be either logical or physical. Storage subsystems, which generally comprise a number of disk drives that can be aggregated and made to appear as virtual disk drives to one or more client computers, usually deploy a cache, which is used to hold frequently accessed disk blocks, to improve input-output performance.

[0021] One or more aspects of the present invention take advantage of the fact that where user data and meta data are separated, the processing of meta data in conjunction with file access provides additional information which can be used to inform a storage subsystem of future input/output (I/O) access requests. This information can be utilized by a storage subsystem to facilitate the management of its internal caches.

[0022] An example of the management of the contents of a cache in a data storage subsystem which utilizes information obtained by processing file meta data in accordance with an aspect of the present invention is described as follows with reference to FIG. 4.

When a client computer 210 wants to read and update some disk blocks associated with a file located in a storage subsystem 230, client computer 210 must be granted an exclusive lock on the associated disk blocks. Client computer 210 initiates a transaction to meta data controller 220, requesting the lock. This lock request indicates that client computer 210 intends to perform I/O operations on a certain range of disk blocks in the future. If meta data controller 220 can grant the requested lock, meta data controller 220 passes a "hint" to storage subsystem 230, which stores those blocks, indicating that the storage subsystem can expect to receive an I/O request from a client computer for a particular range of disk blocks. Meta data controller (MDC) 220 communicates this "hint" to storage subsystem 230 via private network 222. Essentially concurrently, meta data controller 220 grants the lock by signaling to client computer 210 via meta data network 212. In this exemplary embodiment, MDC 220 is an example of a request manager, and storage subsystem controller 236 of storage subsystem 230 is an example of a data object manager.

[0023] If storage subsystem 230 determines that the requested disk blocks are not in cache 232, it pre-fetches the requested blocks from storage disks 234 into cache 232. After receiving the requested lock, client computer 210 initiates an I/O operation with storage subsystem 230 via data network 214 to access at least some of the disk blocks on which a lock was received. When the client-initiated I/O request is received by storage subsystem 230, the storage subsystem may have the requested disk blocks in its cache already as a result of pre-fetching. If not, storage subsystem 230 has already commenced the necessary physical I/O to load the requested blocks into cache 232 as a result of previously receiving a hint from meta data controller 220. When the requested disk blocks are available in cache 232, they are sent to client computer 210 from cache 232 via data network 214. The result of storage subsystem 230 initiating disk input/output in order to store disk blocks that are subject to a future access request by a client computer in cache 232 in advance of receiving a request from client computer 210 is that data access latency is reduced.

[0024] The method of the present invention is also utilized in the operation of the example illustrated in FIG. 4 when a client computer writes disk blocks to the storage subsystem. If client computer 210 sends a transaction to MDC 220, indicating that the client computer has closed a file and has finished writing blocks to the storage subsystem, meta data controller 220 communicates this information to storage subsystem 230 via private network 222. Storage subsystem 230 determines whether to free the storage locations in cache 232 in which the disk blocks comprising the closed file are stored based on this file-closed message and possibly "hints" received regarding other future data access requests. Freeing storage locations in cache 232 permits storage of other disk blocks in cache for expedited access.

[0025] Another example of managing the contents of a cache in a data storage subsystem which utilizes information obtained by processing file meta data in accordance with an aspect of the present invention relates to a computer writing a large file which is not likely to be read. An example of such a file is a checkpoint/restart file, created by a long-running computational job, or a database log file. These files are typically used to recover the state of a computational workload after a computer crash. Since computer crashes are very rare, checkpoint/restart files are typically written regularly, but rarely read. Knowledge of this information can be used to inform a storage subsystem not to cache a checkpoint/restart file once it has been written to disk.

[0026] This example is described further with reference to the exemplary environment of FIG. 4. When computer 210 wishes to write a checkpoint/restart file, it requests write permission from meta data controller 220 via the meta data network 212 and informs meta data controller 220 that the file should not be actively cached. Alternatively, meta data controller 220 could automatically recognize that the file is of a particular type (such as a checkpoint/restart file). Meta data controller 220 grants permission to computer 210 to write the file, providing a list of blocks where the file should be written, and simultaneously, via private network 222, informs storage

controller 236 of storage subsystem 230 that computer 210 is about to write a large file that should not be actively cached in storage subsystem cache 232.

[0027] When computer 210 writes the file via data network 214 to storage subsystem 230, storage subsystem controller 236 decides how much of cache 232 to allocate to storing all or part of the large file and, as quickly as possible, writes the contents of the large file to storage disks 234 within the storage subsystem. As soon as the contents (or partial contents) of the file are written to storage disks 234 of the storage subsystem, the associated file data within the cache 232 can be discarded immediately, since it is highly unlikely that this file will need to be read again. Thus, utilization of the cache based on knowledge of the type of data being stored, which is gained through processing the file's meta data, facilitates optimization of the use of the cache resource.

[0028] An example related to the previous example involves a computer reading a checkpoint/restart file described above to recover the state of a computational workload after the computer has crashed. Knowledge that this type of file is rarely read can be used to inform the storage subsystem to not cache the file when it is being read from disk. It should be noted that the management of the contents of a cache in a data storage subsystem described below with respect to this example applies to reading any large file which is only likely to be accessed infrequently.

[0029] With reference to FIG. 4, computer 210 intends to read a checkpoint/restart file so computer 210 requests permission from meta data controller 220 via meta data network 212 to read the file and informs meta data controller 220 that the file should not be actively cached. Alternatively, meta data controller 220 automatically recognizes that the file is of a particular type (such as a checkpoint/restart file), rather than having to be informed by computer 210. Meta data controller 220 grants permission to computer 210 to read the file, providing a list of blocks where the file is located, and simultaneously, via the private network 222, informs storage subsystem controller 236 of storage subsystem 230 that computer 210 is about to read a large file that should not be actively cached in the storage subsystem cache 232. When computer 210 reads the file via data

network 214 from storage subsystem 230, storage subsystem controller 236 decides how much of cache 232 to allocate to reading the file from storage disks 234. As soon as the contents (or partial contents) of the file are transmitted to computer 210, the associated file data within the cache 232 can be discarded immediately, since it is highly unlikely that this file will need to be read again. Thus, utilization of the cache based on the type of data being accessed, which is determined by processing the file's meta data, frees cache resources to facilitate faster access to other files.

[0030] Another example of an environment using a technique for managing requests for access to data objects to effectuate management of the contents of a cache in a data storage subsystem by utilizing information obtained from processing file meta data in accordance with an aspect of the present invention is described below with reference to FIG. 5. In this example, a number of computers (310, 311, and 312) are members of a database cluster. Each of computers 310, 311, and 312 hosts an instance of the cluster's database. When a computer 310 wants to read and update some disk blocks associated with a database table located in a storage subsystem 330, computer 310 must be granted an exclusive lock on the associated disk blocks. Computer 310 requests a lock on those disk blocks from database lock manager 320 by sending a request on external network 314. If database lock manager 320 can grant the requested lock, database lock manager 320 grants the lock via a message sent to computer 310 on external network 314 and essentially simultaneously sends a message to storage subsystem 330, indicating that the storage subsystem can expect to receive an I/O request from a client computer for a particular range of disk blocks. Database lock manager 320 communicates this message to storage subsystem 330 via private network 322. Analogously to the previous example, database lock manager 320 is an example of a request manager, and storage subsystem 330 comprises a data object manager.

[0031] If storage subsystem 330 determines that the disk blocks for which a lock was granted are not in cache 332, storage subsystem 330 initiates an I/O operation to storage disks 334. Computer 310 initiates an I/O operation with storage subsystem 330 since it

has been granted a lock on the requested disk blocks. When the I/O request initiated by computer 310 is received by storage subsystem 330 via data network 316, the storage subsystem may have already pre-fetched the requested disk blocks and stored them in cache 332. Even if all requested disk blocks have not yet been loaded in the cache, the storage subsystem has initiated the physical I/O from storage disks 334 prior to receiving the request from computer 310. As a result, the latency in providing the requested disk blocks from cache 332 is less than it would be without pre-fetching prompted by the "hint" from the database lock manager.

[0032] In another example of a computer environment embodying the present invention, which is described with reference to FIG. 6, a centralized storage meta data controller 434 is co-hosted with a storage subsystem 420. In this environment, one or more computers are connected to the data storage subsystem via a data network. By way of example, as shown in FIG. 6, a computer 410 exchanges data with a storage subsystem 420 via data network 412. Storage subsystem 420 comprises a server 430 connected to a cache 322 and storage disks 424. Server 430 comprises logical partitions 431 and 432.

[0033] In this example, the functions of meta data controller 434 and storage subsystem controller 433 are executed by software running in logical partitions (LPARs) 432 and 431, respectively, of server 430. Using virtual input/output bus 436 between the meta data controller LPAR 432 and the storage subsystem controller LPAR 431, "hints" regarding anticipated future I/O requests directed to storage subsystem 420 are passed at very high speed and low latency from meta data controller 434 to storage subsystem controller 433. The benefit of pre-fetching disk blocks into the storage subsystem cache is enhanced by the use of high-speed, low-latency communications between the meta data controller and storage subsystem controller. In this example, meta data controller 434 and storage subsystem controller 433 are examples of a request manager and a data object manager, respectively.

[0034] The present invention can be included in an article of manufacture (e.g., one or more computer program products) having, for instance, computer usable media. The

media has therein, for instance, computer readable program code means or logic (e.g., instructions, code, commands, etc.) to provide and facilitate the capabilities of the present invention. The article of manufacture can be included as a part of a computer system or sold separately.

[0035] Additionally, at least one program storage device readable by a machine embodying at least one program of instructions executable by the machine to perform the capabilities of the present invention can be provided.

[0036] The flow diagrams depicted herein are just examples. There may be many variations to these diagrams or the steps (or operations) described therein without departing from the spirit of the invention. For instance, the steps may be performed in a differing order, or steps may be added, deleted or modified. All of these variations are considered a part of the claimed invention.

[0037] Although preferred embodiments have been depicted and described in detail herein, it will be apparent to those skilled in the relevant art that various modifications, additions, substitutions and the like can be made without departing from the spirit of the invention and these are therefore considered to be within the scope of the invention as defined in the following claims.